MAIN FRAME PLUS★WARE™ NEWS
A PUBLICATION OF STSC, INC.
December 1985 Issue 3

Letter To Our Customers

During the month of August, we conducted a survey to determine what products and services are important to running your APL operation. I am very pleased to report excellent returns and would like to take this opportunity to thank you for your prompt and enthusiastic response. Your input will help us plan our future development efforts.

I'd also like to give you the status of some of your requests and suggestions.

Your suggestions for new features and products were particularly valuable. Our development team is presently at work evaluating the technical feasibility of individual interface products, further PC/Mainframe compatibility, full screen utilities, and add-ons and enhancements to APL2. Once this work is complete, it, along with additional market studies, will determine the priorities of these projects.

Most respondents felt that this newsletter was worthwhile. However, a surprising number of people who responded to the survey hadn't received any copies of the newsletter. We have verified and updated our mailing list to ensure that all of our customers receive future issues.

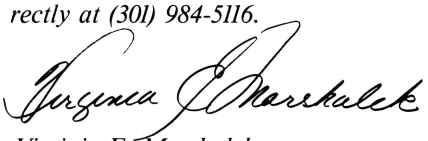
Many people suggested we include more performance tips in the newsletter. As a result of this suggestion, you will see, in this issue and from now on, suggestions for optimizing code and ways to use the FASTFNS utilities provided with the last major release of the APL★PLUS System.

We also received requests to publish articles on what our customers are doing with the APL★PLUS System. We were in fact quite fascinated with the large variety of applications in use. They range from budgeting and accounting systems to robotics. We are very interested in what you have accomplished with the APL★PLUS System and welcome the opportunity to publish articles in this newsletter. We welcome volunteers to send articles directly to me at STSC, 2115 East Jefferson St., Rockville, MD 20852.

You've also suggested that we send out information on the contents of new releases one month prior to sending the tape. This is an excellent idea (wish we'd thought of it) and we will implement it with Release 5.1.

We were pleased to note that the majority of our customers found the current support service "acceptable". However, acceptable isn't good enough. We are currently working to improve the level of support to our customers. You will notice new people taking calls and we hope a more reliable, responsive approach to the way we resolve problems and answer questions.

If I can be of assistance, either with product suggestions or general questions, feel free to call me directly at (301) 984-5116.



Virginia E. Marshalek
Director of Marketing

STSC Releases Shared Files For the APL2 Environment

STSC has released SHAREFILE™/AP, a secure, shared access, component-based APL file system for the APL2 environment. SHAREFILE/AP will use standard IBM interfaces to provide APL data manipulation and storage features without modifications to APL2.

SHAREFILE/AP offers a sophisticated auxiliary processor design, providing the features and benefits you have come to expect from any APL file system. SHAREFILE/AP is designed to take advantage of many of the inherent properties of APL2. For example, the file system will handle *any* APL2 data object, including general or nested array features. Data is not translated or truncated in any way. SHAREFILE/AP also takes full advantage of APL2's event signalling and reporting mechanism. Error messages and completion codes fit right into the APL2 error message scheme.

SHAREFILE/AP also supports international language translations, which can provide you with error messages and command names in languages other than English. A workspace of easy-to-use file functions is provided with SHAREFILE/AP. This workspace allows you to use simple names like FREAD, FAPPEND, and FDROP to access and update your data.

Since SHAREFILE/AP uses a mnemonic protocol to perform file system operations, users who wish to access the shared variable directly can use defined names rather than numeric codes. A single shared variable protocol design allows fast, efficient, and uninterruptable operations because only one assignment of the shared variable is

continued on page 4

Understanding Execution Time

Why are some programs faster than others? A basic understanding of execution time models can eliminate the mystery and help you write more efficient programs.

Due to the interpretive nature of APL, two different models are appropriate for small and large arrays. When applied to small arrays, all primitive APL operations take approximately the same amount of CPU time. This is because each APL operation has an overhead of about 500 machine instructions for interpretation and memory management. As a result, when processing small arrays, CPU time tends to be determined by the number of APL operations executed.

A basic understanding of execution time models can eliminate the mystery of why some programs are faster than others and help you write more efficient programs.

In contrast, when processing large arrays, interpretive overhead can be dwarfed by the time spent processing elements, and CPU time tends to be determined by the number of low-level machine operations required to compute the result. Different APL operations may require vastly different amounts of time to execute. For example, adding two N-element vectors requires (at least) N operations. Computing the outer product of two N-element vectors requires N*2 operations, one for each element of the result.

A common way of characterizing the execution time of an algorithm is the "big-oh" notation. Roughly, a O(N) (read "big-oh N") algorithm is one that executes in time proportional to N; a O(N*2) algorithm executes in time proportional to N squared. Constant factors are ignored for purposes of big-oh comparison. In spite of this, the big-oh rating gives an accurate prediction of an algorithm's performance as N approaches infinity. Given a sufficiently large argument, a O(N) algorithm will be faster than a O(N*2) algorithm, a O(N*2) algorithm will be faster than a O(N*3) algorithm, and so forth. O(N) algorithms are sometimes referred to as "linear"; less efficient algorithms are referred to as "nonlinear".

The big-oh rating of an entire program is just the maximum big-oh rating of operations used in the program. Most primitive APL operations are O(N), where N is the number of elements in the largest argument or result. The big-oh rating of some operations is not obvious. For instance, grade up is usually implemented with a merge sort that is O(N × log N).

As an example of determining big-oh ratings, consider the problem of counting the number of 1's, 2's, 3's, and so forth in a vector. Given a N-element vector of integers in the range 1 to N, the object is to produce an N-element result Z such that Z[1] gives the number of 1's in the vector, Z[2] the number of 2's, and so forth. Three different solutions to this problem are presented in Figure 1.

The first program, COUNT1, simply counts the number of 1's in the argument, the number of 2's, etc. The operations in the program are all O(N) except the outer product and the plus reduction, which are O(N*2). Thus, overall, COUNT1 is O(N*2).

```
     ∇Z←COUNT1 B
[1]   Z←+/(⍳0⌈⌈/B)∘.=B
     ∇


     ∇Z←COUNT2 B;L;W;F
[1]   B←B[⍋B] ⋄ L←B≠1⌽B ⋄ L[(0≠⍴B)/⍴B]←1
[2]   W←0,L/⍳⍴L ⋄ F←(1↓W)-¯1↓W
[3]   Z←(0⌈⌈/B)⍴0 ⋄ Z[L/B]←F
     ∇


     ∇Z←COUNT3 B;I
[1]   Z←(0⌈⌈/B)⍴0 ⋄ I←⍴B ⋄ →(I=0)/0
[2] L1:Z[B[I]]←Z[B[I]]+1
[3]   →(0<I←I-1)/L1
     ∇
```

Figure 1

The second program, COUNT2, sorts the argument, finds the position of the last occurence of each number, and computes the frequencies from those positions. The operations in COUNT2 are all O(N) except for grade up, which is O(N × log N). Thus, COUNT2 is O(N × log N).

The third program, COUNT3, initializes the result to zeroes, then uses each element of the argument as an index to increment the appropriate element of the result. The array operations are O(N). The operations in the loop process only scalars, and the loop executes N times, yielding a O(N) process. Thus, COUNT3 is O(N).

The execution time of the three COUNT programs was measured for arguments of various sizes using the EXT function from the MFFNS workspace. The results are presented in Figure 2.

Relative CPU Time

N	COUNT1	COUNT2	COUNT3	Compiled
1	7	19	14	6
10	10	24	71	8
100	179	50	622	19
1000	15740	357	6327	136
10000	WS FULL	4113	62752	1490

Figure 2

As expected, for small arguments, COUNT1 is fastest because it uses the fewest APL operations. For large arguments, where the big-oh rating is more meaningful, the COUNT2 function is fastest. The COUNT3 function is not fastest, even though it is O(N), because it processes scalars. The big-oh rating hides a large factor in COUNT3's execution time that prevents it from being faster than COUNT2 until the arguments are astronomically huge (about 1E120 elements). However, note that COUNT3 is more than twice as fast as COUNT1 for 1000-element arguments. [The lesson here is that O(N × log N) is not much worse than linear, but O(N*2) is much worse than linear.] The last column gives the execution time for a compiled version of COUNT3. Freed of interpretive overhead, the compiled function reveals the great potential speed of the O(N) COUNT3 algorithm.

continued on next page

From these results, it is clear that it is important to recognize and eliminate unnecessary nonlinear algorithms when optimizing programs. Inner and outer products often signal potential inefficiency. The following statements may produce a O(N*2) program when imbedded in a loop that iterates O(N) times:

```
X←(M[;1]=V[I])/M[;2]
I←N+(N↓V)⍳S
A←I↑V ◇ V←I↓V
Z←Z,A
```

Sometimes, just using different APL operations can eliminate a O(N*2) algorithm. For example, the last two statements above can be sped up by using indexing and indexed assignment. In other cases, it may be necessary to use the APL compiler or FASTFNS. The compiler makes it possible to efficiently implement algorithms which involve detailed processing of scalars. The FASTFNS workspace in library 3 contains precompiled utility functions that may be used in place of some common nonlinear operations. For instance, *MATIOTA* may be used in place of `∧.=`, and *ORETRIEVE* and *NEXTIN* may be used in place of the first two expressions above. Documentation for these and other precompiled functions is available on-line in the FASTFNS workspace. The *EXT* function, which is used to measure execution time, is documented in the MFFNS workspace in library 3.

The compiler makes it possible to efficiently implement algorithms which involve detailed processing of scalars.

Meet The Team

In this issue of *Mainframe PLUS★WARE News,* we'll introduce the developer of SHAREFILE/AP, Patrick A. Parks.

Patrick A. Parks

Patrick has been with STSC more than four years and has more than seven years of programming experience in APL. In addition to APL, he's experienced in FORTRAN, BASIC, Assembly, PL/I, PASCAL, and C. Patrick became familiar with STSC's APL★PLUS System many years ago at UCLA's Department of Medicine while using APL to develop medical diagnosis models. He joined STSC shortly thereafter to solve business problems for customers, optimize customer applications, and participate in the development of STSC products like DATAPORT.

Developing the first shared file system for the APL2 environment may be one of Patrick's most ambitious projects to date. The task involved developing an auxiliary processor (AP) to link STSC's SHAREFILE System with APL2. The process took more than nine months and included developing installation procedures, testing the system, and developing new documentation for end users and system administrators.

Early in the project, Patrick determined that building the AP as a "shell" that manipulates data, evaluates results, and performs generic data handling would give STSC the flexibility to add new functionality to SHAREFILE in the future. What this also means is that as APL2 users realize the need for additional function and feature beyond what APL2 provides, we can meet their needs with simple, straightforward solutions in a quick and efficient manner without building a new AP.

Part of the fun of developing SHAREFILE/AP was being able to take advantage of general arrays to implement a shared variable protocol using a single variable. This technique allows greater efficiency than traditional protocols.

Patrick's future plans involve adopting SHAREFILE/AP to the MVS environment and collecting feedback and responding to the needs of the APL2 community as they request additional features and add-ons to the product.

As an APL programmer himself, Patrick says he feels particularly fortunate to be involved in the development of tools that improve the productivity of APL programmers. He also noted that "working in the company of other dedicated and enthusiastic people like those at STSC encourages me to design and implement products of superior quality, with an emphasis on getting the job done right".

Patrick is married and the proud father of a baby boy, with another one on the way. He works out of his home in Canoga Park, California.

MAIN FRAME PLUS★WARE™ NEWS
A PUBLICATION OF STSC, INC.

Mainframe PLUS★WARE News is published by STSC, Inc. and is distributed free of charge to the company's customers. Comments regarding the newsletter should be sent to:

Mainframe Newsletter Editor
STSC, Inc.
2115 East Jefferson St.
Rockville, MD 20852

©1985 STSC, Inc. All rights reserved. Printed in U.S.A.

APL★PLUS is a registered trademark of STSC, Inc.
PLUS★WARE and SHAREFILE are trademarks of STSC, Inc.

Tama Traberman Joins STSC To Provide APL Educational Services

Tama Traberman, Ed.D., a well-known teacher, author, and speaker in the APL community, has joined STSC to develop and conduct APL training and educational services.

Dr. Tama Traberman

Dr. Traberman plans to implement a full curriculum of APL classes at the introductory, intermediate, and advanced levels. Seminars and workshops on special topics, such as techniques for using STSC's software on personal computers and in mainframe environments, will also become a part of the educational program. Classes and seminars will be offered at STSC's headquarters in Rockville, Maryland; at STSC's New York City office; in several large U.S. cities as needs dictate; and at STSC customer sites, by arrangement.

A member of the New York/SIGAPL Board of Directors, Dr. Traberman has been active in promoting that group's annual professional development seminar, "APL—A Tool For Thought". She gave the keynote speech at the 1984 SIGAPL Professional Development Seminar and, most recently, was an invited luncheon speaker at the 1985 International Conference on APL (APL85) in Seattle, Washington.

A survey is enclosed with this issue of the newsletter, giving you the opportunity to provide Dr. Traberman with your training requirements and specific areas of interest. To ensure that you are on the mailing list to receive announcements, be sure to complete and return the survey or call Dr. Traberman directly at (201) 751-9305.

SHAREFILE/AP

continued from page 1

required to perform an operation. You need not worry about spurious file damage due to the possibility of the resumption of incomplete file operations.

Additional features and benefits of SHAREFILE/AP make it a valuable investment for the APL application developer:

- SHAREFILE/AP supports very large files and can handle vast quantities of data with no degredation in performance. Each individual file can contain over two gigabytes of data, and there is no practical limit on the number of files the system can contain.
- Files are easily shared with other users, even concurrently. SHAREFILE/AP permits many users to read and update the same file simultaneously without delays or damage to data.
- SHAREFILE/AP offers exceptional security and file control. The creator of a file has exclusive access by default; others can access the file only if the owner grants explicit permission. The owner has a fine degree of control over use of his files, letting, for example, others append data to a file but not read or modify the contents in any other way.

For more information on SHAREFILE/AP, contact an APL★PLUS Mainframe Software representative.

MAIN FRAME PLUS★WARE™ NEWS

STSC, Inc.
2115 East Jefferson Street
Rockville, Maryland 20852

Return Postage Guaranteed

BULK RATE
U.S. POSTAGE
PAID
PERMIT No. 4590
ROCKVILLE, MD 20850